

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN 2  
IMAGE PROCESSING

GV hướng dẫn : Nguyễn Ngọc Toàn  
Sinh viên thực hiện : Lê Thị Hồng Hạnh  
MSSV : 22127103  
Lớp : 22CLC07

# Mục lục

<b>I Mô tả các hàm</b>	<b>2</b>
1 Thay đổi độ sáng . . . . .	2
2 Thay đổi độ tương phản . . . . .	3
3 Lật ảnh . . . . .	4
4 Chuyển đổi ảnh RGB sang ảnh Grayscale/Sepia . . . . .	5
5 Làm mờ/sắc nét ảnh . . . . .	6
6 Cắt ảnh theo kích thước ở trung tâm . . . . .	9
7 Cắt ảnh theo khung . . . . .	10
8 Phóng to/thu nhỏ ảnh . . . . .	13
9 Main . . . . .	14
<b>II Test case</b>	<b>15</b>
1 Thay đổi độ sáng . . . . .	15
2 Thay đổi độ tương phản . . . . .	16
3 Lật ảnh . . . . .	16
4 Chuyển đổi ảnh RGB sang ảnh Grayscale/Sepia . . . . .	17
5 Làm mờ/sắc nét ảnh . . . . .	17
6 Cắt ảnh theo kích thước ở trung tâm . . . . .	17
7 Cắt ảnh theo khung . . . . .	18
8 Phóng to/thu nhỏ ảnh . . . . .	18
<b>III Đánh giá mức độ hoàn thành</b>	<b>19</b>
<b>IV Tài liệu tham khảo</b>	<b>20</b>

# I Mô tả các hàm

## 1 Thay đổi độ sáng

### Ý tưởng

Ta có giá trị của mỗi điểm ảnh RGB là 3 kênh màu trong đó giá trị của từng kênh màu nằm trong khoảng [0, 255], màu đen có giá trị (0, 0, 0) và màu trắng có giá trị (255, 255, 255). Như vậy, để tăng độ sáng cho ảnh ta chỉ việc cho các giá trị của điểm ảnh tiến đến gần giá trị màu trắng hơn bằng cách cộng cách giá trị đó cho một số beta.

### Mô tả hàm

```
def adjust_brightness(img, beta)
```

Input:

- img (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)
- beta (int): số nguyên được cộng vào giá trị của mỗi điểm ảnh

Output:

- new\_img (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Cộng beta cho giá trị ở 3 kênh màu của mỗi điểm ảnh
- Mỗi kênh màu chỉ có giá trị nằm trong khoảng [0, 255], việc cộng cho một số nguyên có thể dẫn đến giá trị nằm ngoài khoảng trên. Do đó, ta dùng hàm numpy.clip() để giới hạn giá trị của mỗi kênh màu trong khoảng [0, 255]

## 2 Thay đổi độ tương phản

### Ý tưởng

Dộ tương phản thể hiện sự khác biệt giữa hai màu đen và trắng trên một bức ảnh, độ tương phản càng cao thì khoảng khác biệt này càng lớn. Để tăng độ tương phản cho ảnh, ta cần làm cho các điểm ảnh tối càng tối hơn và các điểm ảnh sáng sẽ càng sáng hơn. Do đó, ta cần tăng sự khác biệt này bằng cách nhân giá trị mỗi điểm ảnh cho một số alpha.

### Mô tả hàm

```
def adjust_contrast(img, alpha)
```

Input:

- img (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)
- alpha (int): số nguyên được nhân vào giá trị của mỗi điểm ảnh

Output:

- new\_img (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Nhân alpha cho giá trị ở 3 kênh màu của mỗi điểm ảnh
- Mỗi kênh màu chỉ có giá trị nằm trong khoảng [0, 255], việc nhân cho một số nguyên có thể dẫn đến giá trị nằm ngoài khoảng trên. Do đó, ta dùng hàm numpy.clip() để giới hạn giá trị của mỗi kênh màu trong khoảng [0, 255]

### 3 Lật ảnh

#### Ý tưởng

Mỗi bức ảnh là một ma trận có kích thước (height, width, channels) nên lật ảnh chỉ là việc lật ma trận theo chiều ngang hoặc dọc, numpy có hỗ trợ các hàm cho việc lật ma trận.

#### Mô tả hàm

```
def flip_horizontal(img)
def flip_vertical(img)
```

Input:

- img (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)

Output:

- new\_img (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Để lật ma trận theo chiều ngang, sử dụng hàm numpy.fliplr() với đối số truyền vào là ma trận gốc và hàm trả về ma trận được lật ngang
- Để lật ma trận theo chiều dọc, sử dụng hàm numpy.flipud() với đối số truyền vào là ma trận gốc và hàm trả về ma trận được lật dọc

## 4 Chuyển đổi ảnh RGB sang ảnh Grayscale/Sepia

### Ý tưởng

- Grayscale: Sử dụng luminosity method để chuyển 3 giá trị kênh màu (R, G, B) của một điểm ảnh thành một giá trị duy nhất với công thức:

$$\text{Grayscale} = 0.3 * R + 0.59 * G + 0.11 * B$$

- Sepia: Gần giống như chuyển sang ảnh grayscale, ta cần thay đổi các 3 giá trị kênh màu của điểm ảnh thành 3 giá trị kênh màu mới với công thức:

$$\text{newRed} = 0.393 * R + 0.769 * G + 0.189 * B$$

$$\text{newGreen} = 0.349 * R + 0.686 * G + 0.168 * B$$

$$\text{newBlue} = 0.272 * R + 0.534 * G + 0.131 * B$$

### Mô tả hàm

```
def RGB_to_grayscale(img)
def RGB_to_sepia(img)
```

Input:

- img (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)

Output:

- new\_img (numpy.ndarray): ma trận ảnh đầu ra

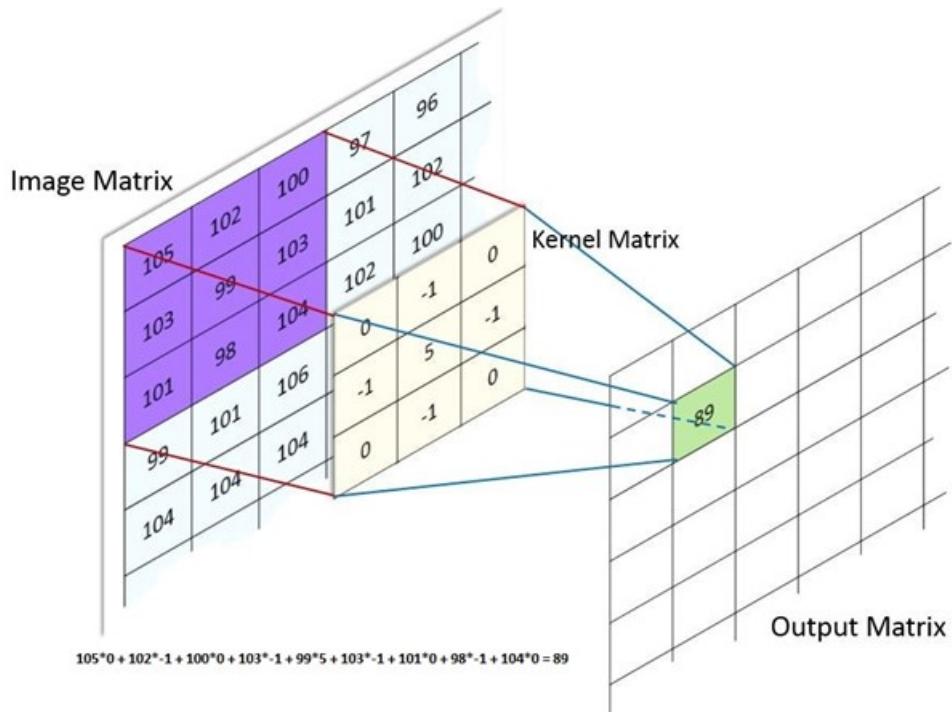
Description:

- Để thực hiện tính toán, ta tạo ma trận mới bằng ma trận đầu vào với kiểu dữ liệu của các phần tử là float.
- Sử dụng list comprehension lấy các giá trị kênh màu của ma trận đầu vào để tính toán theo công thức nêu trên và tạo ma trận đầu ra có kích thước tương tự.
- Chuyển tất cả giá trị của ma trận đầu ra thành kiểu dữ liệu int.
- Mỗi kênh màu chỉ có giá trị nằm trong khoảng [0, 255], việc tính toán có thể dẫn đến giá trị nằm ngoài khoảng trên. Do đó, ta dùng hàm numpy.clip() để giới hạn giá trị của mỗi kênh màu trong khoảng [0, 255].

## 5 Làm mờ/sắc nét ảnh

### Ý tưởng

Sử dụng tích chập (convolution) với một ma trận bộ lọc kernel để tính toán lại toàn bộ giá trị kenh màu của điểm ảnh trên ma trận ảnh gốc. Phép tích chập được thực hiện bằng cách dịch chuyển ma trận kernel lần lượt qua tất cả các phần tử của ma trận gốc, trong đó điểm giữa của kernel được đặt tương ứng với điểm đang xét trên ma trận gốc. Phần tử đang xét sẽ được tính toán bằng cách nhân element-wise kernel với ma trận phụ được cắt từ ma trận gốc có kích thước bằng kernel sau đó tính tổng.



Trong đó, ma trận kernel thường được sử dụng cho làm mờ ảnh là:

$$gaussian\_kernel\_3 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$gaussian\_kernel\_5 = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Ma trận kernel được sử dụng làm sắc nét ảnh là:

$$sharpen\_kernel = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Đối với những điểm ảnh ở rìa ma trận, ta gặp khó khăn khi tính tích chập vì khi xem điểm đó làm trung tâm thì không tìm được giá trị xung quanh nó. Do đó, trước khi thực hiện tích chập,

ta tạo một ma trận thêm padding từ ma trận gốc với độ dày của padding này bằng kích thước kernel chia 2 (kernel luôn là ma trận vuông và kích thước lẻ).

Lấy ví dụ với ma trận đầu vào A có kích thước  $2 \times 2$  và sử dụng gaussian\_kernel\_3 làm bộ lọc, ta sẽ tính được ma trận B như sau:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \text{ padding} \rightarrow A' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B[0][0] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} \times \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{13}{16}$$

$$B[0][1] = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 2 & 1 & 0 \end{bmatrix} \times \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{7}{8}$$

$$B[1][0] = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \times \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{7}{8}$$

$$B[1][1] = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{13}{16}$$

$$\Rightarrow B = \begin{bmatrix} \frac{13}{16} & \frac{7}{8} \\ \frac{7}{8} & \frac{13}{16} \end{bmatrix}$$

## Mô tả hàm

```
def blur(img)
def sharpen(img)
```

Input:

- img (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)

Output:

- new\_img (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Khi bắt đầu lấy kernel, nếu làm mờ ảnh thì chọn gaussian\_kernel được đề cập ở trên, còn nếu làm sắc nét ảnh thì kernel là sharpen\_kernel.
- Từ kernel, ta tính được độ dày cho padding và tạo ma trận mới từ ma trận gốc có thêm lớp padding xung quanh là những phần tử có giá trị 0. Ta gọi ma trận này là padding\_img.
- Duyệt qua từng phần tử của ma trận gốc, giá trị mới tại phần tử đó là kết quả tích chập của ma trận con được slicing từ padding\_img với phần tử đang xét là trung tâm và kernel. Khi nhân hai ma trận, ma trận con của padding\_img có kích thước (kernel\_height, kernel\_width, 3) và kernel có kích thước (kernel\_height, kernel\_width) nên không thể thực hiện phép nhân.

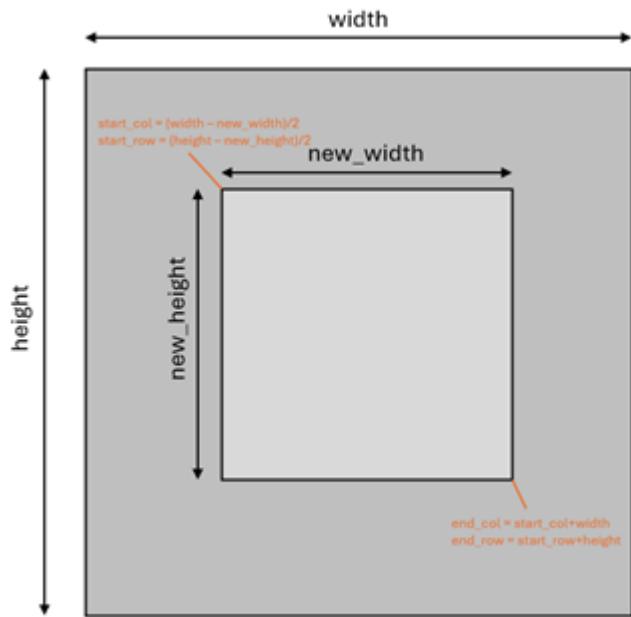
Ta phải thêm một chiều vào kernel bằng `numpy.newaxis`, điều này cho phép nhân hai ma trận bằng broadcasting. Kết quả của phép nhân thu được ma trận kích thước (`kernel_height`, `kernel_width`, 3), dùng hàm `numpy.sum()` với `axis=(0,1)` để tính tổng của mỗi kênh màu trong ma trận đó và lưu vào ma trận kết quả.

- Mỗi kênh màu chỉ có giá trị nằm trong khoảng [0, 255], việc tính toán có thể dẫn đến giá trị nằm ngoài khoảng trên. Do đó, ta dùng hàm `numpy.clip()` để giới hạn giá trị của mỗi kênh màu trong khoảng [0, 255].

## 6 Cắt ảnh theo kích thước ở trung tâm

### Ý tưởng

Từ kích thước mới, ta có thể tính được tọa độ điểm bắt đầu và kết thúc của ảnh mới. Phép chia 2 là để cắt ảnh giữa chiều dài và chiều rộng ảnh. Sau đó chỉ cần sử dụng kỹ thuật slicing từ ma trận gốc để có được ma trận kết quả.



### Mô tả hàm

```
def crop_center(img, crop_size)
```

Input:

- `img` (numpy.ndarray): ma trận ảnh gốc kích thước (`height`, `width`, `channels`)
- `crop_size` (tuple): kích thước cần crop (`new_height`, `new_width`)

Output:

- `new_img` (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Từ kích thước của ma trận đầu vào và kích thước ảnh cần crop, ta tính được điểm chỉ số hàng, cột của điểm bắt đầu và kết thúc của ảnh sau khi crop.
- Sử dụng slicing để tạo ảnh kết quả.

## 7 Cắt ảnh theo khung

### Ý tưởng

- Chung: Ta cần tìm tọa độ điểm giữa ảnh làm tâm của hình tròn hoặc elip, sau đó tìm phương trình cho đường tròn hoặc elip xác định các điểm nằm ngoài khung và cho giá trị điểm ảnh tại đó là 0. Kết quả ảnh cho ra cuối cùng chỉ giữ lại màu của các điểm ảnh nằm trong khung, các điểm nằm ngoài sẽ được chuyển thành màu đen.

- Khung tròn:

Từ kích thước ảnh gốc tính được bán kính  $R$  của hình tròn, sử dụng phương trình đường tròn có tâm là tọa độ điểm giữa ảnh. Phương trình gốc là của đường tròn với  $(i, j)$  là tọa độ tâm:

$$(x - i)^2 + (y - j)^2 = R^2$$

Các điểm nằm trong đường tròn sẽ thỏa:

$$(x - i)^2 + (y - j)^2 \leq R^2$$

- Khung elip chéo nhau:

Ta có phương trình elip có tâm tại  $(i, j)$  là:

$$\frac{(x - i)^2}{a^2} + \frac{(y - j)^2}{b^2} = 1$$

Elip ở đây là 2 elip nằm chéo nhau, do đó ta cần thực hiện xoay elip 1 góc 45 độ ( $\frac{\pi}{4}$  radian) và 135 độ ( $\frac{3\pi}{4}$  radian). Phương trình tổng quát cho elip quay quanh tâm O một góc  $\alpha$ :

$$\frac{(x \cos \alpha - y \sin \alpha)^2}{a^2} + \frac{(x \sin \alpha + y \cos \alpha)^2}{b^2} = 1$$

Tuy nhiên, ta cần tìm  $a, b$  sao cho hai hình elip chéo nhau tiếp xúc vào các cạnh của ảnh. Ta tiến ảnh phân tích khi  $\alpha = \frac{\pi}{4}$ :

$$\begin{aligned} & \frac{(x \cos \frac{\pi}{4} - y \sin \frac{\pi}{4})^2}{a^2} + \frac{(x \sin \frac{\pi}{4} + y \cos \frac{\pi}{4})^2}{b^2} = 1 \\ & \Leftrightarrow \frac{(\frac{\sqrt{2}}{2}x - \frac{\sqrt{2}}{2}y)^2}{a^2} + \frac{(\frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y)^2}{b^2} = 1 \\ & \Leftrightarrow \frac{(x - y)^2}{2a^2} + \frac{(x + y)^2}{2b^2} = 1 \\ & \Leftrightarrow \frac{x^2 - 2xy + y^2}{2a^2} + \frac{x^2 + 2xy + y^2}{2b^2} = 1 \\ & \Leftrightarrow (\frac{1}{2a^2} + \frac{1}{2b^2})x^2 - (\frac{1}{a^2} - \frac{1}{b^2})xy + (\frac{1}{2a^2} + \frac{1}{2b^2})y^2 - 1 = 0 \end{aligned}$$

Giả sử elip tiếp xúc cạnh hình vuông tại điểm có tung độ  $y=k$ .

$$(1) \Leftrightarrow (\frac{1}{2a^2} + \frac{1}{2b^2})x^2 - k(\frac{1}{a^2} - \frac{1}{b^2})x + (\frac{1}{2a^2} + \frac{1}{2b^2})k^2 - 1 = 0$$

Ta nhận thấy (1) là phương trình Parabol có  $a = (\frac{1}{2a^2} + \frac{1}{2b^2})$ ,  $b = -k(\frac{1}{a^2} - \frac{1}{b^2})$ ,  $c = (\frac{1}{2a^2} + \frac{1}{2b^2})k^2 - 1$ . Để elip chỉ tiếp xúc với cạnh hình vuông thì (1) phải có nghiệm duy nhất, điều kiện để phương trình bậc 2 có nghiệm duy nhất là:

$$\begin{aligned}\Delta &= b^2 - 4ac = 0 \\ \Leftrightarrow k^2(\frac{1}{a^4} - \frac{2}{a^2b^2} + \frac{1}{b^4}) - 4(\frac{1}{2a^2} + \frac{1}{2b^2})(\frac{k^2}{2a^2} + \frac{k^2}{2b^2} - 1) &= 0 \\ \Leftrightarrow \frac{k^2}{a^4} - \frac{2k^2}{a^2b^2} + \frac{k^2}{b^4} - \frac{k^2}{a^4} - \frac{k^2}{a^2b^2} + \frac{1}{a^2} - \frac{k^2}{a^2b^2} - \frac{k^2}{b^4} + \frac{1}{b^2} &= 0 \\ \Leftrightarrow -\frac{4k^2}{a^2b^2} + \frac{1}{a^2} + \frac{1}{b^2} &= 0 \\ \Leftrightarrow \frac{-4k^2 + 2a^2 + 2b^2}{a^2b^2} &= 0 \\ \Leftrightarrow a^2 + b^2 &= 2k^2\end{aligned}$$

Khoảng cách từ tâm của elip đến tung độ  $y=k$  là một nửa chiều dài ảnh. Đặt kích thước cạnh hình vuông là  $\text{side}=2k$ , khi đó:

$$a^2 + b^2 = \frac{1}{2}\text{side}^2$$

Ta có thể chọn  $a, b$  tùy ý sao cho  $a>b$ , cụ thể chọn  $a=3b$  ta có:

$$a^2 = \frac{3}{4} \cdot \frac{\text{side}^2}{2}, b^2 = \frac{1}{4} \cdot \frac{\text{side}^2}{2}$$

Kết luận, ta có phương trình elip tâm  $(i, j)$  quay quanh tâm một góc  $\alpha$  và tiếp xúc với các cạnh hình vuông có độ dài  $\text{side}$  là:

$$\frac{((x-i)\cos\alpha - (y-j)\sin\alpha)^2}{\frac{3}{4} \cdot \frac{\text{side}^2}{2}} + \frac{((x-i)\sin\alpha - (y-j)\cos\alpha)^2}{\frac{1}{4} \cdot \frac{\text{side}^2}{2}} = 1$$

Các điểm nằm trong elip sẽ thỏa:

$$\frac{((x-i)\cos\alpha - (y-j)\sin\alpha)^2}{\frac{3}{4} \cdot \frac{\text{side}^2}{2}} + \frac{((x-i)\sin\alpha - (y-j)\cos\alpha)^2}{\frac{1}{4} \cdot \frac{\text{side}^2}{2}} \leq 1$$

## Mô tả hàm

```
def crop_circle(img)
def crop_ellipse(img)
```

Input:

- $\text{img}$  (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)

Output:

- $\text{new\_img}$  (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Khung tròn:

Tìm tọa độ tâm ảnh (j,i) vì index trong ma trận là (row, col) và độ dài bán kính. Khởi tạo một ma trận mask có kích thước (height, width) để lưu giá trị luận lý xác định điểm ảnh tại tọa độ đó có nằm trong khung hay không. Ta có:  $mask[i][j] = \begin{cases} True & \text{if } img[i][j] \text{ in circle} \\ False & \text{if } img[i][j] \text{ not in circle} \end{cases}$

Sử dụng numpy.ogrid[:img.shape[0], :img.shape[1]] để tạo giá trị tọa độ theo chiều dài và rộng của ma trận đầu vào.

Giá trị luận lý của  $mask[y][x] = True$  nếu  $(x - i)^2 + (y - j)^2 \leq R^2$  với (j, i) là tọa độ tâm của hình tròn.

Ma trận kết quả là bản sao của ma trận đầu vào và các tọa độ có  $mask[y][x] = False$  thì các kênh màu sẽ nhận giá trị 0.

- Khung elip chéo nhau:

Tìm tọa độ tâm ảnh (j,i) vì index trong ma trận là (row, col) và chiều dài hình vuông. Khởi tạo một ma trận mask\_1 và mas\_2 có kích thước (height, width) để lưu giá trị luận lý xác định điểm ảnh tại tọa độ đó có nằm trong khung hay không.

$$mask\_1[i][j] = \begin{cases} True & \text{if } img[i][j] \text{ in ellipse at 45 degree angle} \\ False & \text{if } img[i][j] \text{ not in ellipse at 45 degree angle} \end{cases}$$

$$mask\_2[i][j] = \begin{cases} True & \text{if } img[i][j] \text{ in ellipse at 135 degree angle} \\ False & \text{if } img[i][j] \text{ not in ellipse at 135 degree angle} \end{cases}$$

$$mask = mask\_1 \| mask\_2$$

Sử dụng numpy.ogrid[:img.shape[0], :img.shape[1]] để tạo giá trị tọa độ theo chiều dài và rộng của ma trận đầu vào.

Giá trị luận lý của  $mask\_1[y][x] = True$  nếu  $\alpha = \frac{\pi}{4}$ ,  $mask\_2[y][x] = True$  nếu  $\alpha = \frac{3\pi}{4}$  và thỏa:

$$\frac{((x - i) \cos \alpha - (y - j) \sin \alpha)^2}{\frac{3}{4} \cdot \frac{side^2}{2}} + \frac{((x - i) \sin \alpha - (y - j) \cos \alpha)^2}{\frac{1}{4} \cdot \frac{side^2}{2}} \leq 1$$

Ma trận kết quả là bản sao của ma trận đầu vào và các tọa độ có  $mask = False$  thì các kênh màu sẽ nhận giá trị 0.

## 8 Phóng to/thu nhỏ ảnh

### Ý tưởng

Việc thu nhỏ hay phóng to ảnh chính là tăng hoặc giảm số lượng điểm ảnh trên ảnh gốc mà giá trị tại điểm ảnh trên ảnh đầu ra sẽ được lấy từ giá trị điểm ảnh nào đó trên ảnh đầu vào. Sử dụng phương pháp Nearest neighbor interpolation hay còn gọi là pixel replication để thực hiện zoom ảnh. Như tên gọi của phương pháp, chúng ta chỉ sao chép các điểm ảnh trên ảnh gốc vào ảnh đầu ra. Gọi factor là chỉ số phóng to hoặc thu nhỏ,  $\text{factor} = \text{kích thước ảnh đầu ra} / \text{kích thước ảnh đầu vào}$ . Giá trị điểm ảnh đầu ra tại tọa độ  $(i, j)$  sẽ được sao chép giá trị tại tọa độ  $(i/\text{factor}, j/\text{factor})$  của ảnh đầu vào. Ảnh sẽ mất đi độ sắc nét nếu phóng to hoặc thu nhỏ quá đà.

10	4	22
2	18	7
9	14	25

10	10	4	4	22	22
2	2	18	18	7	7
2	2	18	18	7	7
9	9	14	14	25	25
9	9	14	14	25	25

angeljohnsy.blogspot.com

### Mô tả hàm

```
def zoom(img, factor)
```

Input:

- img (numpy.ndarray): ma trận ảnh gốc kích thước (height, width, channels)
- factor (float): bằng 2 khi phóng to 2x, bằng 0.5 khi thu nhỏ 2x

Output:

- new\_img (numpy.ndarray): ma trận ảnh đầu ra

Description:

- Từ ma trận đầu vào và factor, ta xác định được kích thước ma trận đầu ra và khởi tạo tạm thời cho nó giá trị 0 toàn ma trận.
- Duyệt qua mỗi điểm ảnh trên ma trận đầu ra, tìm điểm ảnh tương ứng trên ma trận gốc bằng và copy giá trị tại điểm ảnh đó vào ảnh đầu ra  $\text{new\_img}[i, j, :] = \text{img}[\text{int}(i/\text{factor}), \text{int}(j/\text{factor}), :]$

## 9 Main

### Mô tả hàm

- Có 9 options thao tác với ảnh, người dùng nhập lựa chọn từ 1 đến 8 để chọn thao tác mong muốn, chọn 0 nếu muốn thực hiện tất cả chức năng.

```
Enter img_path:/content/Lenna.png
0. All
1. Adjust brightness
2. Adjust contrast
3. Flip
4. Convert to grayscale/sephia
5. Blur/Shapern
6. Crop in center
7. Crop in circle/ellipse frame
8. Zoom in/out 2x
```

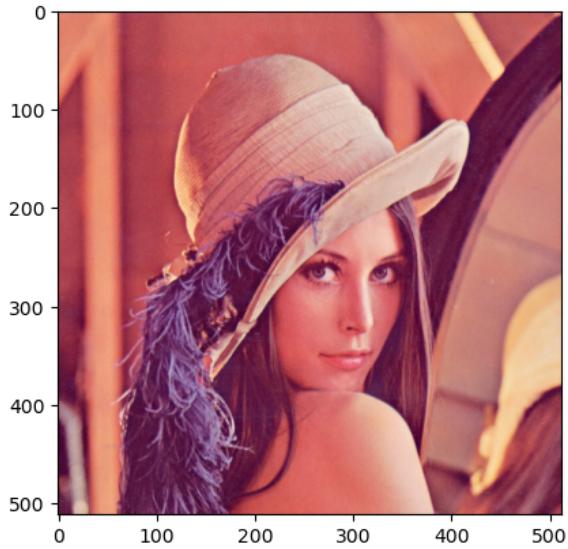
- Đối với một số chức năng có yêu cầu nhập thêm đối số, người dùng nhập đối số truyền vào hàm để hàm thực thi. Các hàm đó là: thay đổi độ sáng, thay đổi độ tương phản, crop ảnh ở trung tâm.
- Sau khi thực hiện các chức năng, ảnh được in ra và lưu vào file đầu ra tương ứng với các chức năng

## II Test case

Ảnh đầu vào (input)

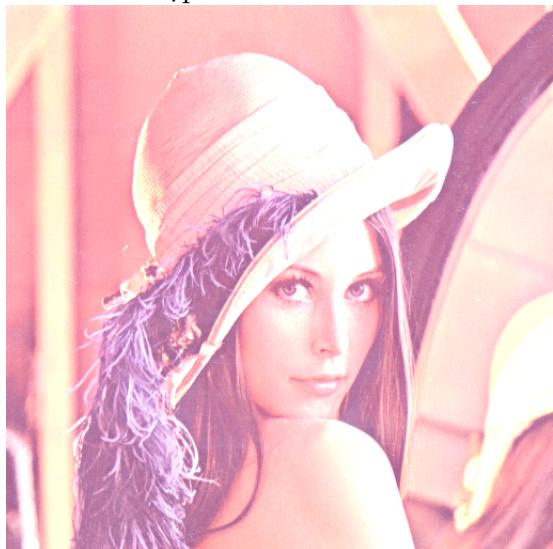


Kích thước ảnh ban đầu (512x512)



### 1 Thay đổi độ sáng

Với beta nhập vào là: 100



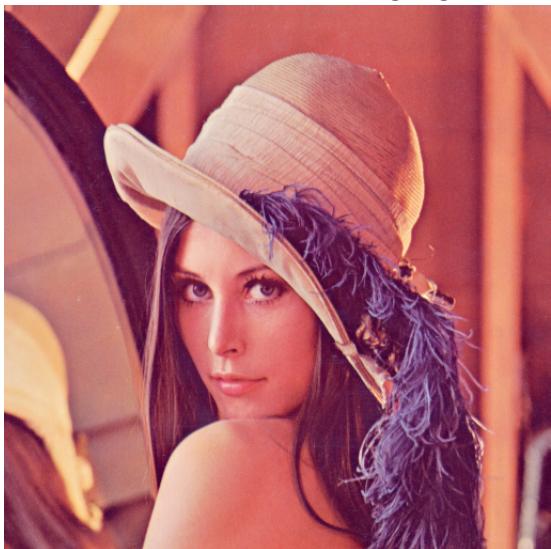
## 2 Thay đổi độ tương phản

Với alpha nhập vào là: 1.5

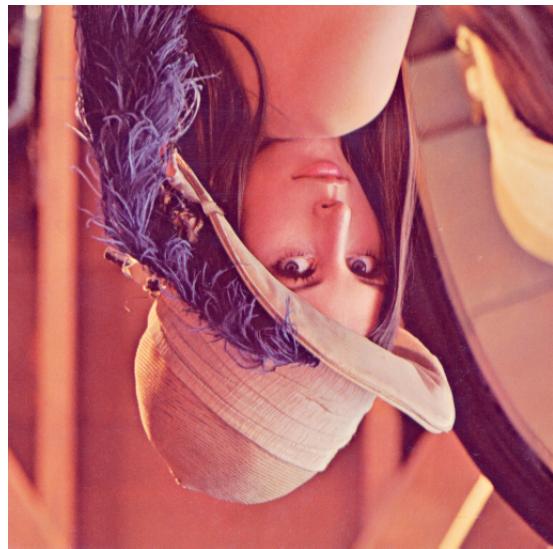


## 3 Lật ảnh

Lật ảnh theo chiều ngang



Lật ảnh theo chiều dọc



#### 4 Chuyển đổi ảnh RGB sang ảnh Grayscale/Sepia

Grayscale

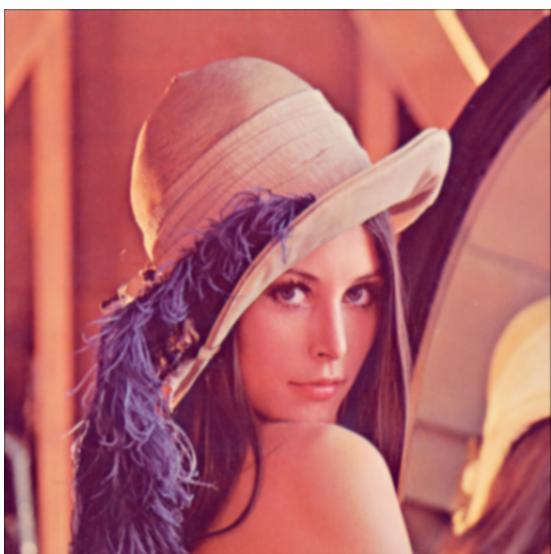


Sepia

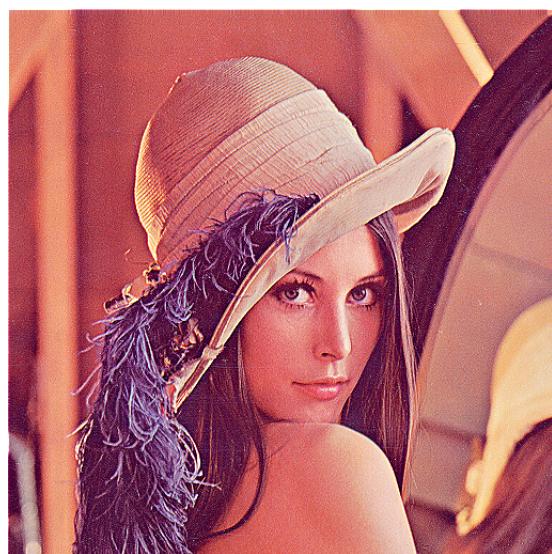


#### 5 Làm mờ/sắc nét ảnh

Làm mờ



Làm sắc nét



#### 6 Cắt ảnh theo kích thước ở trung tâm

Với chiều dài mới là 250 và chiều rộng mới là 250



## 7 Cắt ảnh theo khung

Khung tròn

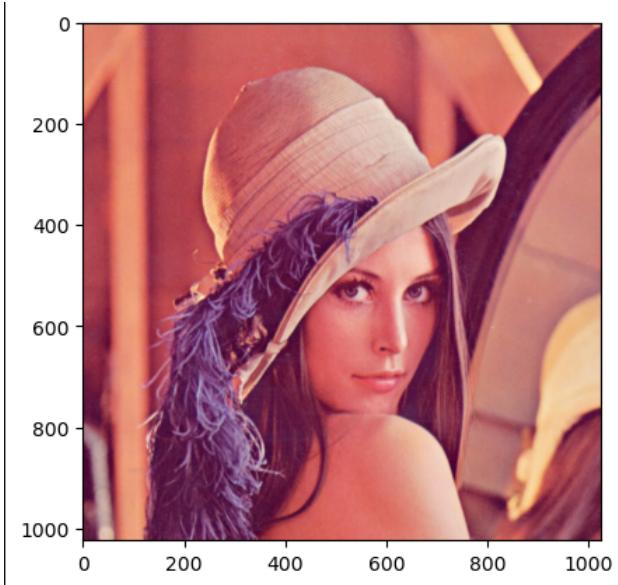


Khung elip chéo nhau

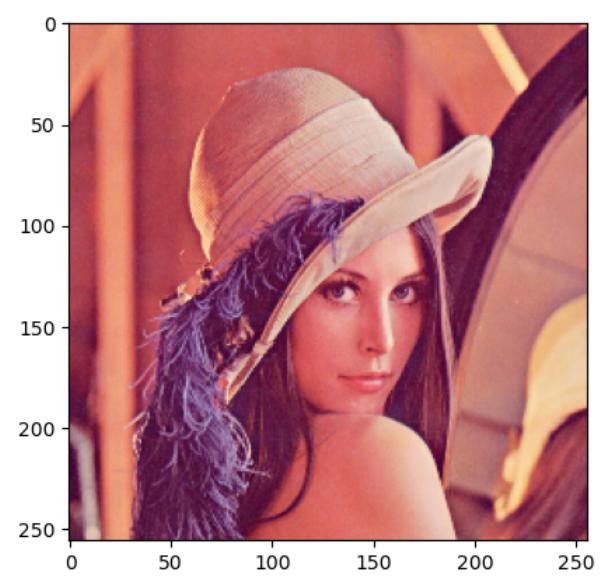


## 8 Phóng to/thu nhỏ ảnh

Phóng to 2x



Thu nhỏ 2x



### III Đánh giá mức độ hoàn thành

STT	Chức năng/Hàm	Mức độ hoàn thành	Ảnh kết quả
1	Thay đổi độ sáng	100%	Ở phần Test case (II)
2	Thay đổi độ tương phản	100%	
3.1	Lật ảnh ngang	100%	
3.2	Lật ảnh dọc	100%	
4.1	RGB thành Grayscale	100%	
4.2	RGB thành Sepia	100%	
5.1	Làm mờ ảnh	100%	
5.2	Làm sắc nét ảnh	100%	
6	Cắt ảnh theo kích thước	100%	
7.1	Cắt ảnh theo khung tròn	100%	
7.2	Cắt ảnh theo khung elip	100%	
8	Hàm main	100%	
9	Phóng to/thu nhỏ 2x	100%	

## IV Tài liệu tham khảo

Brightness and Contrast:

- [https://docs.opencv.org/4.x/d3/dc1/tutorial\\_basic\\_linear\\_transform.html](https://docs.opencv.org/4.x/d3/dc1/tutorial_basic_linear_transform.html)

Flip:

- <https://numpy.org/doc/stable/reference/generated/numpy.flip.html>

Grayscale and Sepia:

- <https://www.baeldung.com/cs/convert-rgb-to-grayscale>
- <https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image>

Blur and Sharpen:

- [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))
- <https://github.com/khanhnhan1512/Image-Processing>

Circle and Ellipse:

- <https://stackoverflow.com/questions/44865023/how-can-i-create-a-circular-mask-for-a-numpy-array>
- <https://github.com/khanhnhan1512/Image-Processing>

Zoom in/out:

- [https://www.tutorialspoint.com/dip/zooming\\_methods.htm](https://www.tutorialspoint.com/dip/zooming_methods.htm)
- <https://www.youtube.com/watch?v=v9CFu4r6tPY&t=622s>